Full name: AYUSH ASHNA  
Department: ROS  
Join date: 6 June 2022  
Week 1 Assignment (heading): Learning about ROS

Aim: To learn about ROS, and use it on a Ubuntu-based computer with Python

Components Used: ROS:

ROS stands for Robot Operating System. It is essentially the ‘software’ part of robotics. It is used to communicate, visualise, analyse data and run simulations relevant to the concerned robot. I learned the basics of it in Python, although C++ is supported as well.

Day 1-3:

Mostly spent on reading the book called “Programming Robots with ROS” by Morgan Quigley, Brian Gerkey & William D. Smart. It covers the basics of the system, how ROS works. It describes how to set up ROS workspace via catkin, how to set up nodes and create subscribers and publishers, service and clients and using messages. It also described the code in sufficient detail.

Also, I was told to try out some tutorials on the [ROS wiki](http://wiki.ros.org/ROS/Tutorials) to get some on hand experience with the software. This was communicated by Mr. Dhanush about how much practise I should do.

I had to first install Ubuntu on Windows and then after that I had to install ROS, so it took a lot of time.

Important code snippets:

workspace\_folder/ -- WORKSPACE

src/ -- SOURCE SPACE

CMakeLists.txt -- 'Toplevel' CMake file, provided by catkin

package\_1/

CMakeLists.txt -- CMakeLists.txt file for package\_1

package.xml -- Package manifest for package\_1

...

package\_n/

CMakeLists.txt -- CMakeLists.txt file for package\_n

package.xml -- Package manifest for package\_n

This describes a general catkin workspace created using catkin packages. Every package inside has a package.xml and CMakeLists.txt file inside it. Creating these directories involves very basic linux directory commands.

Day 4-5:

I completed all the tutorials on the Beginner level and five on the Intermediate level. All of them were done via catkin (not rosbuild) in Python, though using the alternatives isn’t much hassle as their tutorials are very similar. I learned on how to create a node and topic, and what they are. But I also learned some of the most important basics in ROS - how to communicate between nodes. There are two ways: Publishers and Subscribers, Service and Client

Publishers and Subscribers:

Nodes are pieces of code used to do stuff in ROS. Topics are like buses over which nodes exchange data via messages. Nodes publish message on a topic, which other nodes which are subscribed to the topic can read. ROS Master is used as a central link between nodes, it tells which node is called what and registers the name for better communication.

I will include a code snippet describing this:

pub = rospy.Publisher(’topic\_name', String, queue\_size=10)

rospy.init\_node('name\_of\_publihser', anonymous=True)

This is part of a publisher’s code. It describes that the node name\_of\_publihser is publishing to topic topic\_name

Similar to this, the subscriber node is used to listen to a topic.

rospy.init\_node('name\_of\_subscriber', anonymous=True)

rospy.Subscriber("topic\_name", String, callback)

This essentially means that the subscriber name\_of\_subscriber named is subscribing to topic\_name

essentially mea

The “anonymous=True” is so that ROS generates a unique name for this node by adding some random numbers at the end, so that nodes with same name don’t cause an issue.

Serive and Client:

Similar to publishers, a node can offer a service to other nodes that need that service, clients. However, there is a feedback system that offers two way communication between, unlike Publishers and Subscribers. The client sends a message as a request and the service node provides, along with a feedback message of “Yes, I got it and here’s what I did”.

rospy.Service('myservice', CustomService, ServiceFunction)

Here, the service provided is myservice and the service type CustomService, which is user defined. The ServiceFunction is invoked to handle the information and then can print the feedback on the console.

#1 rospy.wait\_for\_service('myservice')

#2 rospy.ServiceProxy(‘myservice’,CustomService)

#3 resp1 = myservice(\_parameters)

1. The client requests for service named myservice
2. “We create a handle for calling this service”- line straight from Wiki
3. After that, the handle is essentially a normal function

There are many other smaller things, but this is the thing that struck to me as most important.

There were also some minor tutorials on a turtle simulation, which didn’t amount to any modification.